Introduction to SDN

Portions of this PPT draw from PPT authored by Professor Dijiang Huang at Arizona State University

Outline

- **Concept of SDN**
- OpenFlow – a SDN Implementation

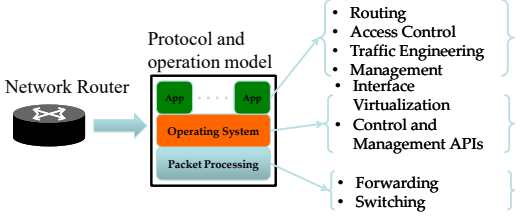
What is SDN?

- Software-Defined Networking, a.k.a. the programmable network
- A new network architecture
- A new network operating system
- A Generalized network virtualization
- *(more…)*

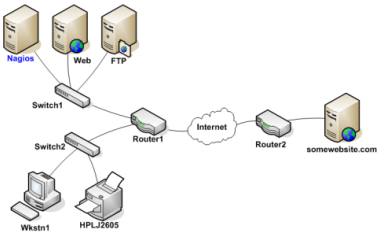
Why do we need SDN?

- Do you think a Network system simple?
- Networks used to be simple: Ethernet, IP, TCP….
- New **control** requirements led to great complexity
  - Isolation ➔ VLANs, ACLs
  - Traffic engineering ➔ MPLS, ECMP,Weights
  - Packet processing ➔ Firewalls, NATs, middleboxes
  - Payload analysis ➔ Deep packet inspection (DPI)
  - …..
- Mechanisms designed and deployed independently
  - Complicated "control plane" design, primitive functionality

Networking Devices

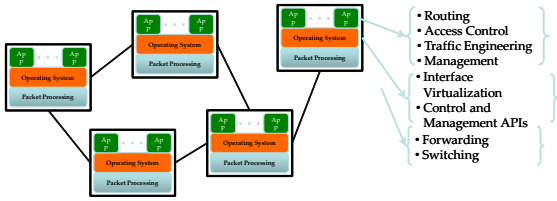


A Typical Network



Today's Distributed Networking Systems

- Close Boxes, Fully Distributed Protocols



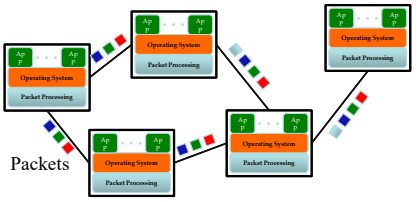
Closed Operating Environment

- Traffic control decisions are mostly made **locally**, i.e., usually require a negotiation procedures among network entities (e.g., routing).
- Not easy for testing and experiments, i.e., usually requires a separate testing environments.

7

Data processing and Control in Distributed Systems

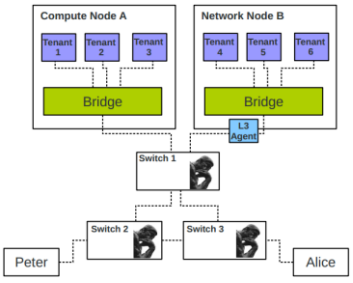
Data processing and controls are **not separated**. Algorithms, e.g., routing, is performed in a distributed environment. Each router acts **individually**, much likes our current social networks.



8

Switched Networks in OpenStack

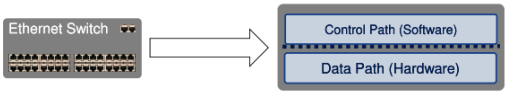
- Switches **learn** from the network traffic they observe and **decide independently**.



9

Traditional network node: Switch

- Two "planes"
  - **Control Plane**: computing the forwarding state
    - Involves coordination with rest of system
  - **Data Plane**: forwarding packets
    - Based on local forwarding state



10

Two "planes"

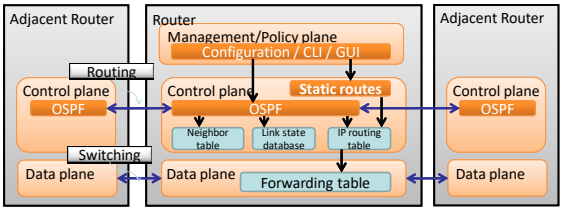
- Two fundamental terms to begin understanding the SDN

Processing Plane	Where it runs	How fast these processes run	Type of processes performed
Control Plane	Switch CPU (smart but slow)	In the order of thousands of packets per second	Routing protocols (i.e. OSPF, IS-IS, BGP), Spanning Tree, SYSLOG, AAA (Authentication Authorization Accounting), NDE (Netflow Data Export), CLI (Command Line interface), SNMP
Data Plane	Dedicated Hardware ASIC (fast but dumb)	Millions or Billions of packets per second	Layer 2 switching, Layer 3 (IPv4 | IPv6) switching, MPLS forwarding, VRF Forwarding, QOS (Quality of Service) Marking, Classification, Policing, Netflow flow collection, Security Access Control Lists

11

Traditional network node: Router

- Router can be partitioned into control and data plane
  - Management plane/ configuration
  - Control plane / Decision: OSPF (Open Shortest Path First)
  - Data plane / Forwarding



12

SDN definitions

- SDN is an approach to building computer networks that separates and abstracts elements of these systems – *from Wikipedia*
  - In SDN, not all processing happens inside the same device.
- In the Software Defined Networking architecture, the control and data planes are **decoupled**, network intelligence and state are **logically centralized**, and the underlying network infrastructure is **abstracted** from the applications.
  – *from ONF White Paper*

13

How to do it?

- How to get a simpler, more systematic design for the so complicate network control mechanisms?
- The power of Abstraction
  - "Modularity based on abstraction is the way things get done." – Barbara Liskov
- **Abstractions ➔ Interfaces ➔ Modularity**

14

Finding control plane abstractions

- Task: Compute forwarding state:
  - Consistent with low-level hardware/software
    - which might depend on particular vendor
  - Based on entire network topology
    - because many control decisions depend on topology
  - For all routers/switches in network
    - every router/switch needs forwarding state

15

SDN Abstractions

- SDN is defined *precisely* by three abstractions: Forwarding, Distribution, Configuration
  - Abs#1: Be compatible with low-level hardware/software
    - Need an abstraction for general **forwarding model**
  - Abs#2: Make decisions based on entire network
    - Need an abstraction for **distributed network state**
  - Abs#3: Compute the configuration of each physical device
    - Need an abstraction that **simplifies configuration**

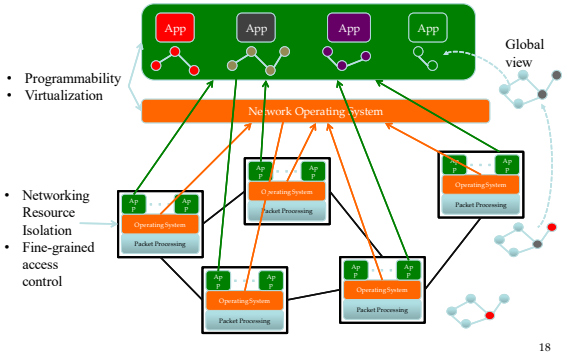
16

Abstraction Detail

- Abs#1: ***Forwarding abstraction***
  - **OpenFlow** is current proposal for forwarding standard
  - Configuration in terms of **flow entries**: <header, action>
- Abs#2: ***Network state abstraction***
  - **Global network view** abstraction
  - **Network OS** (controllers) queries network devices to form "view" and sends commands to them to control forwarding
- Abs#3: ***Specification abstraction***
  - Abstract view of network: simple model with only enough to specify goals.
  - **Network virtualization**: map abstract configuration to physical configuration

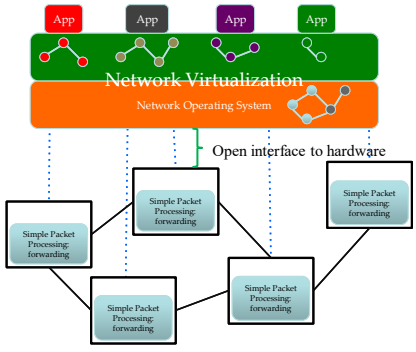
17

From traditional networking to SDN

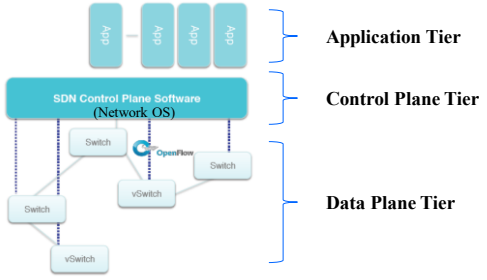


18

Abstract SDN Model



What SDN looks like



Clean Separation of Concerns

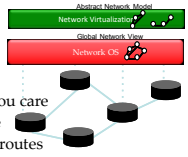
- **Control program:** express goals on abstract view
  - Driven by **Operator Requirements**
- **Virtualization Layer:** abstract view ←→ global view
  - Driven by **Specification Abstraction** for particular task
- **NOS:** global view ←→ physical switches
  - API: driven by **Network State Abstraction**
  - Switch interface: driven by **Forwarding Abstraction**

Software-Defined Networking

- Data Plane Tier
  - Packet forwarding (as per flow table), packet manipulation (as per flow table), statistics collection
- Control Plane Tier (Network OS)
  - Data plane resource marshaling, common libraries (e.g., topology, host metadata, state abstractions)
- Application Tier
  - Virtual network overlays, network slicing (delegation), tenant-aware broadcast, application-aware path computation, integration with other software packages, policy, security, traffic engineering.

How to process the SDN requests?

- Write a simple program to configure a simple model
  - Configuration merely a way to specify what you want
- Examples
  - ACLs: who can talk to who
  - Isolation: who can hear my broadcasts
  - Routing: only specify routing to the degree you care
    - Some flows over satellite, others over landline
  - TE: specify in terms of quality of service, not routes



- Virtualization layer "compiles" these requirements
  - Produces suitable configuration of actual network devices
- NOS then transmits these settings to physical boxes

Outline

- Concept of SDN
- **OpenFlow – a SDN Implementation**

OpenFlow

- OpenFlow is a Layer 2 communications protocol that gives access to the forwarding plane of a network switch or router over the network.
- OpenFlow enables controllers to determine the path of network packets through the network of switches.

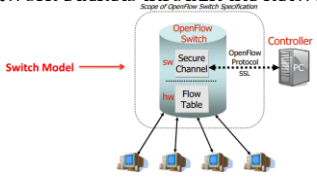
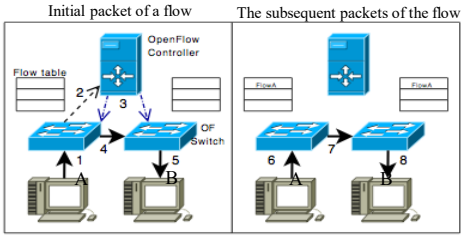


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

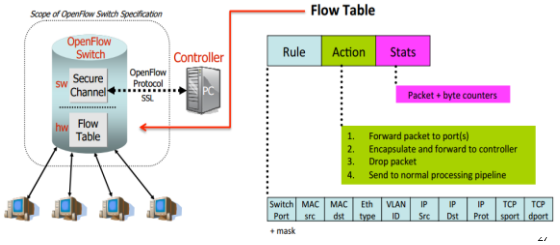
How it works?

- The OpenFlow protocol is layered on top of the Transmission Control Protocol (TCP), and prescribes the use of Transport Layer Security (TLS).
- Controllers should listen on TCP port 6653 for switches that want to set up a connection. Earlier versions of the OpenFlow protocol unofficially used port 6633.

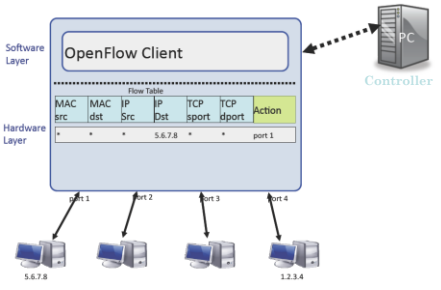


OpenFlow Switch, v 1.0

- OpenFlow allows remote administration of a layer 3 switch's packet forwarding tables, by adding, modifying and removing packet matching rules and actions.



OpenFlow Flow Table Abstraction



Flow entry examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching (Efficient !!!)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

OpenFlow Header v.1.1

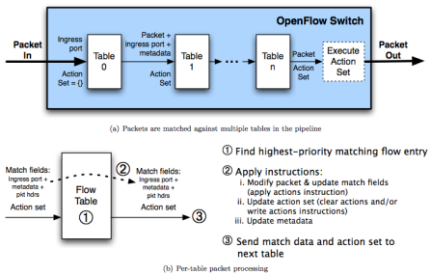
Ingress Port	Metadata	Ether src	Ether dst	Ether type	VLAN id	VLAN priority	MPLS label	MPLS traffic class	IPv4 src	IPv4 dst	IPv4 proto / ARP opcode	IPv4 ToS bits	TCP/ UDP / SCTP src port	ICMP Type	TCP/ UDP / SCTP dst port	ICMP Code

Table 3: Fields from packets used to match against flow entries.

OpenFlow Switch

- OpenFlow-compliant switches come in two types:
  - **OpenFlow-only**: support only OpenFlow operation, in those switches all packets are processed by the OpenFlow pipeline, and can not be processed otherwise.
  - **OpenFlow-hybrid**: support both OpenFlow operation and normal Ethernet switching operation, i.e. traditional L2 Ethernet switching, VLAN isolation, L3 routing (IPv4 routing, IPv6 routing...), ACL and QoS processing.
  - The **OpenFlow pipeline** of every OpenFlow switch contains multiple flow tables, each flow table containing multiple flow entries.

OpenFlow matching process



ONF, "*OpenFlow Swith Specification v.1.3.0*", June, 2012

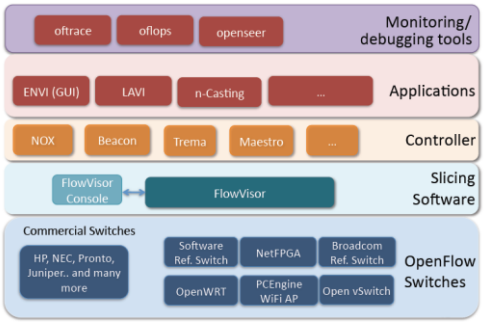
Programmability

- OpenFlow native programmability
  - Redefined the flow format, actions, and etc.
- Controller programmability
  - Call Openflow provided API to control

OpenFlow Controllers

Name	Language	Platform(s)	Original Author	Notes
Openflow Reference	C	Linux	Stanford/Nicira	Not designed for extensibility
Nox/Pox	Python, C++	Linux	Nicira	Actively developed
Beacon	Java	Win, Mac, Linux, Android	David Erickson (standford)	Runtime modular, web UI framework, regression test framework
Trema	Ruby, C	Linux	NEC	Includes emulator, regression test framework
Ryu	Python	Linux	NTT	Supports openstack and openflow 1.3
Floodlight	Java	any	BigSwitch	Support openstack and java based

SDN Stack



Example of OpenFlow Applications

Topic	Demo
Network Virtualization	FlowVisor
Hardware Prototyping	OpenPipes
Load Balancing	PlugNServe
Energy Savings	ElasticTree
Mobility	MobileVMs
Traffic Engineering	Aggregation
Wireless Video	OpenRoads

References

1. Scott Shenker, Software Defined Networking, http://inst.eecs.berkeley.edu/~ee122/
2. OpenFlow official website at http://www.openflow.org
3. NOX controller at http://www.noxrepo.org/
4. J. Pettit, J. Gross, B. Pfaff, M. Casado, S. Crosby, "Virtual Switching in an Era of Advanced Edges," 2nd Workshop on Data Center – Converged and Virtual Ethernet Switching (DC-CAVES), ITC 22, Sep. 6, 2010.
5. B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, S. Shenker, "Extending Networking into the Virtualization Layer," HotNets-VIII, Oct. 22-23, 2009.
6. S. Zhou, "Virtual Networking," ACM SIGOPS Operating Systems Review, 2010.
7. N. Mckeown et al., "OpenFlow: Enabling Innovation in Campus Networks," white papers. The OpenFlow Switch Consortium, March 2008, http://www.openflowswitch.org//documents/openflow-wp-latest.pdf
8. Y. Luo et al., "Accelerating OpenFlow Switching with Network Processors," Proc. ACM/IEEE Symp. Architectures for Networking and Communications Systems (ANCS 09), ACM Press, 2009.
9. J. Naous et al., "Implementing an OpenFlow Switch on the NetFPGA platform," Proc. ACM/IEEE Symp. Architectures for Networking and Communications Systems (ANCS 08), ACM Press, 2008.
10. H. Chen et al., "A Survey on the Application of FPGAs for Network Infrastructure Security," IEEE Communications Surveys & Tutorials, June 2010.
11. F. Azmandian et al., "Virtual Machine Monitor-Based Lightweight Intrusion Detection," ACM SIGOPS Operating Systems Review, July 2011.
12. H. Bos and K. Huang, "A network intrusion detection system on ixp1200 network processors with support for large rule sets," in Technical Report 2004-02. Leiden University, 2004.